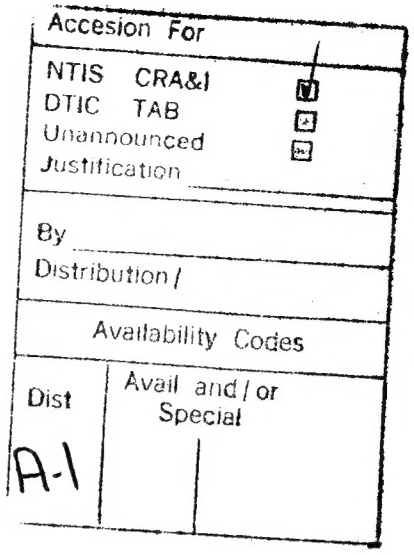
SBIR PHASE I FINAL REPORT

Application of Parallel Processing Technology in Complex Helicopter Analysis

Project Summary

Complex Helicopter Analysis codes, such as the Second Generation Comprehensive Helicopter Analysis System (2GCHAS), have been developed to provide a common framework for interdisciplinary design and analysis of rotorcraft systems. A major limitation on the effectiveness of the second generation comprehensive analysis codes is the slow processing speed that results from the computational intensity of the analysis software. The goal of phase I effort is to carefully define the specification for the parallelization opportunities in 2GCHAS, and apply the parallel processing techniques on a sample analysis to demonstrate a proof of concept. The Onyx Computer System from Silicon Graphics Inc. was chosen as the target machine for this effort as it has 4 processors and was available to ART. The tasks performed during the Phase I Base effort went beyond meeting the Phase I objectives and included the following: 1) Port of the most recent version of 2GCHAS Analysis to the Silicon Graphics Computer System. 2) Parallelize multicase analysis in 2GCHAS. 3) Choose an appropriate analysis component for parallelization in a single case analysis. 4) Parallelize Linearization Analysis and thereby reduce the overall runtime of a single case Linear System analysis. 5) Identify candidates for future parallelization in 2GCHAS. In addition, efforts had been made to improve the eigenanalysis solution method. The result obtained from the Phase I effort showed over an order of magnitude reduction in the overall runtime for the sample problems due to parallelization, better RDB implementation, less overhead in a process invocation, improved analysis algorithm and faster machine speed. The Phase I effort not only showed the technical feasibility of the parallel processing techniques application to a complex helicopter analysis software, but also rendered a port of a significant portion of the 2GCHAS Software which will provide a platform to apply the parallel processing techniques to the other candidate areas in the 2GCHAS Software during the Phase II effort. Moreover, Distributed parallel computing approach presented during the Phase I, will be utilized to couple other related applications such as Computational Fluid Dynamics (CFD) with the Computational Structural Dynamics (CSD) codes such as 2GCHAS to obtain optimal performance.



Contents

List of Figures

Chapter 1

Introduction

Although the first generation helicopter analysis simulations such as GENHEL, REXOR, C81 and CAMRAD, have relatively fast turn around, they do not provide enough capabilities for today's helicopter engineers. Since a decade ago, the ARMY has made a considerable effort to develop a comprehensive helicopter analysis code to analyze performance, aeroelastic stability, loads vibration, aerodynamics, acoustic, stability and control characteristics of rotorcraft. The Second Generation Comprehensive Helicopter Analysis System (2GCHAS), is the result of this effort. In addition to the capabilities of the first generation codes, second generation simulations use finite element methodology for structural dynamics, vortex wake for inflow and Computational Fluid Dynamics (CFD) for aerodynamics.

A major limitation on the effectiveness of the second generation comprehensive analysis codes is the slow processing speed that results from the computational intensity of the analysis software. During the last four years considerable effort has been made to improve the processing speed by modifying existing algorithms or using new ones. The effort resulted in orders of magnitude reduction in run time. Still, with the implementation of CFD and the increase in the size of the today's problems, the current run time is not short enough to be cost effective for a comprehensive helicopter analysis.

Considering that the algorithmic modifications for run time reduction are reaching their optimum level, and the demand for the larger and more CPU intensive problems in the rotorcraft analysis is increasing, new approaches must be considered to improve the run time efficiency in today's helicopter analysis code. One approach is to utilize parallel processing technology.

In parallel processing, the maximum efficiency is reached if all the processors are loaded evenly with the tasks that have little or no data dependency and require minimal synchronization amongst themselves.

Fortunately, comprehensive analysis tools by their nature consist of variety of computationally intensive techniques and algorithms, and present an excellent opportunity

for the parallel processing technology application. For example, multicase analysis where typically the same analysis is executed for a slightly different input in each case, is an ideal candidate for parallelization. Similarly, after a trim or a periodic solution, typically linearization of the periodic coefficient system is performed at several azimuths to obtain an average linearized system before running a linear system analysis. In this case as well, the linearization analysis at each azimuth is fairly independent and presents a ideal opportunity for parallelization.

The Onyx computer system from Silicon Graphics Inc. (SGI) was chosen as the target machine to demonstrate the proof of concept of the application of parallel processing technology to 2GCHAS because it had 4 processors and was already available to ART. However, since 2GCHAS was not available on a SGI platform, a limited port of 2GCHAS to SGI platform was performed during the Phase I activity. In addition, the multicase analysis and the linearization analysis of a periodic system was parallelized to demonstrate the proof of concept.

Chapter 2

Project Objectives

Since 2GCHAS is the most comprehensive helicopter analysis tool and is easily available, it is used as an example to demonstrate the parallel processing opportunities in a complex helicopter analysis system. The general objective of the Phase I effort is to analyze the parallel processing opportunities, to design algorithms for various levels of parallelization and to demonstrate the improvement in the run time efficiency at the highest level of parallelization, namely multicase, and also analyze various system analyses to choose an appropriate candidate for parallelization to reduce runtime in a single case analysis. The following were the Phase I objectives:

1. Port 2GCHAS Analysis Software on SGI - Since the the Onyx Computer system from Silicon Graphics Inc. (SGI) had been chosen as the target machine for the activity, a 2GCHAS analysis port on SGI is required as 2GCHAS is not available on a SGI platform.

2. Develop a Prototype to Perform a Multicase Analysis - Under this objective, the outermost analysis loop, which is executed for as many times as there are input sets for the specified case variables, is parallelized using the multiprocessor structure available in the Onyx Computer System from SGI.

3. Choose an Appropriate Candidate for Parallelization in a Single Case Analysis - Under this objective, various system analysis such as, Trim, Maneuver, and Linear System analyses, will be analyzed to choose an appropriate candidate for parallelization to reduce runtime in a single case analysis.

4. Parallelize Selected System Analysis - Under this objective, a Prototype is to be developed to parallelize the selected analysis to improve runtime in a single case analysis.

5. Identify Other Potential Areas for Parallelization in 2GCHAS - Under this objective, the various system analyses will be carefully analyzed to identify areas for future parallel processing application in 2GCHAS.

Chapter 3

Technical Background

3.1 Comprehensive Helicopter Analysis Background

A complex rotorcraft analysis consists of the following major steps:

1. Input the required analysis data
2. Perform the requested analyses
3. Output the analyses results

Since, the Analysis step is the most CPU intensive, the parallelization effort must be focused to reduce the analysis runtime. In 2GCHAS, the major analyses are Trim, Maneuver and Stability Analysis. In a typical run, one or more of these analyses are run repeatedly for various values of the same set of parameters. This is called a multicase run. In each case one or more of these analyses are executed. Since the Maneuver analysis uses Trim analysis output, Trim and Maneuver analyses may not be run in parallel. The same is true for Trim and Stability analyses.

In performing either a trim, maneuver or linearization in stability analysis, assembly of equations of motions is required. The matrices of all the elements of the model are calculated and then are assembled in system matrices. The assembly process takes a significant amount of CPU time as it is nested inside the trim convergence, periodicity convergence and Newton-Raphson iteration loops. (See Figure 1, 2 and 3 for details). Any improvement in run time of the assembly process would result in significant improvement in Trim, Maneuver and Stability analysis. Incidentally, the assembly process is quite amenable to parallelization.

In Stability analysis, linearization is performed at each azimuth. Once the trimmed values of all states have been determined as a function of azimuth, the linearization of a periodic system at any azimuth does not depend on any other azimuth. This

presents an ideal opportunity for parallelization. In linearization analysis, each state (x, dx, ddx, u) is perturbed, and partial derivatives are formed. (See Figure 4 for details.) Again there is no data dependency between various perturbation in forming the partial derivatives. Consequently, parallelization may also be exploited at each state perturbation level.

The airload and induced velocity computations are performed on each aerodynamic computation point (ACP) and are passed back to the corresponding finite element. On a given iteration, the airloads and induced velocities at each ACP are independent and can therefore be computed in parallel.

The CFD codes are also CPU intensive due to a large number of grid points required to model the flow field for the rotor blade. In fact, according to the latest studies resulting from the efforts to integrate CFD and CSD, 95% of the CPU was consumed by the CFD analysis while only 5% of the CPU was spent in the CSD analysis. Incidentally, a multi- machine parallelization paradigm will be suitable for such an analysis combination.

3.2 Parallel Processing Background

Several parallel computer architectures are available today. Six main classes of parallel processors can be identified as follows:

1. Simple uniprocessor computer, single-instruction, single-data: The processor interprets a single instruction stream to operate on data stored in a single memory. Some parallelism can be achieved through the use of a fetch/decode/execute pipeline.

2. Pipeline multiple-execution-unit uniprocessor for scalar and vector processing: Single- and multiple-statement execution architectures. The processor interprets a single machine instruction to operate, in parallel, on multiple data (e.g., several numbers or a vector) stored in a single memory. In addition, several instructions from the stream also may execute at one time.

3. Lockstep processor: One instruction-execution unit, many processing-element/data-memory pairs. A single machine instruction controls the simultaneous execution in a large number (greater than 100) of processing elements on a lockstep basis. Each processing element has an associated data memory. For this reason, the organization is often called a single-instruction/multiple-data (SIMD) architecture. Intercommunication is via paths among the processing elements.

4. Multiprocessor, Many instructions-execution units operating on many data memories: Each simple uniprocessor accesses programs and data stored in shared memory. Onyx is an example of a multiprocessor.

5. Multicomputer, Many instructions-execution units, each with a dedicated data memory: Each primitive element is a computer (processor-memory pair). Communication among computers is either via fixed paths or via some message switching mechanism.

6. Dataflow architecture: Many instruction-execution units, which are activated upon receipt of data. The order of execution depends on when data is received by the execution units, not on the order in which instructions appear in the source program.

A pipeline uniprocessor can overlap processing of several instructions, attaining a degree of parallelism of from two to four. The multiple-execution-unit uniprocessor increases parallelism to the order of four to ten by simultaneously operating on several instructions from the instruction stream with its different execution units. The lockstep processor uses a simple instruction decoder to process many data items in parallel. Typically, the processing elements are 1 bit wide and together can operate on bits across 100 or more words at the same time.

A multiprocessor consists of many autonomous processors that address the same primary memory. In contrast, multicomputers are made up of autonomous computers communicating by means of messages through static or dynamic communication links. Since a multiprocessor can emulate a multicomputer with a message system implemented in shared memory, the multiprocessor is a more general structure for a distributed computer research laboratory.

Another way to define the place of multiprocessor systems in the world of distributed computers is to consider the synchronization granularity or frequency of synchronization between tasks in a distributed system. Table 1 shows, for a variety of synchronization granularities, the best-suited distributed computer organization. Multiprocessor systems, such as Onyx, are most suitable for a medium to coarse grain of synchronization.

Grain Size	Synchronization interval (instructions)	Distributed computer structure	Communication overhead (instructions)
Fine	1	Vector or array processor	1
Medium	10-100	Multiprocessor	1-10
Coarse	100-10,000	Multicomputer	100-10,000
Large	10,000-10 million	Network	10,000-10 million

Table 1: Synchronization Granularity and Distributed Computer Structure

3.3 2GCHAS Background

The Second Generation Comprehensive Helicopter Analysis System (2GCHAS) is a unique software tool for the development and analysis of comprehensive mathematical models of helicopters. The program is based on a library of primitive modeling elements that allows the user to build complex models by appropriate interconnection of these modeling elements. 2GCHAS is organized into an Executive Complex and a Technology Complex. The Executive Complex provides the operating environment utilized by the Technology Complex in processing the simulated system model. The Technology Complex is organized into six Computer Program Configuration Items (CPCI's). They are as follows:

1) Element Library (EL) This CPCI provides the primitive modeling elements used to support the development of comprehensive helicopter models. These elements include linear and nonlinear beam elements, spring, damper, rigid body mass, and rigid blade elements for modeling structural components. It also provides transfer function elements for modeling simple control systems.

2) Airload and Induced Velocity (ALIV) This CPCI computes the aerodynamic loads and induced velocities acting on structural elements based on a grid of aerodynamic computation points. The aerodynamic grid is specified by the user.

3) Assemble and Solve (AS) The Assemble process combines the equations of each of the modeling elements with the applied constraints to provide a fully coupled model of the complete system for use in the solution process. The Solution process is based on a Newton Raphson iteration process. The states of the modeling elements are adjusted iteratively using a gradient technique until the aerodynamic, structural, and inertial forces are in balance.

4) Linear Systems Analysis (LS) This CPCI provides an extensive linear analysis capability for the comprehensive nonlinear model. A linearized model is used as a starting point to provide Floquet analysis, model order reduction, eigenanalysis, and transfer function generation. Both periodic and constant coefficient linear analysis are supported.

5) Technology Input Processor (TIP) This CPCI provides the user interface to configuring a 2GCHAS model. Through the TIP the user specifies the model configuration and the analysis parameters to be used. Extensive validation facilitates the data entry process. In addition, TIP transforms user input in the form required by the analysis CPCIs.

6) Technology Output Processor (TOP) This CPCI provides the user interface for presenting the output in standard plots or in engineering report form. TOP extracts analysis data from the Saved Calculations (SC) segment of the Run Data Base (RDB). Multicase-multijob plots and reports are available from TOP.

The individual CPCI's may communicate with each other directly through argument

lists in the subroutine calling sequence or they may utilize the Run Data Base, a shared memory object designed to provide convenient access to all essential model data at the global level. The Executive Complex provides interaction with the RDB and the host computer's operating system. 2GCHAS is currently operational on the VAX computer under the VMS operating system and on the SUN workstation under the UNIX operating system.

Chapter 4

Parallelization Approaches

The parallelization in 2GCHAS code can be achieved at several levels using multiprocessor architecture of a single computer system, or multiple computer systems over the network. The proposed approaches are based on the fact that the majority of the elapsed time is spent during the analysis loop. In addition, any reasonable helicopter analysis requires several analysis cases to be run in order to obtain a meaningful output. Therefore, parallelization of the code for the multicase analysis is highly desired. To maximize the resources available for the effort, the following parallelization approaches at the multicase analysis level are proposed for the Phase I:

- Distributed parallel computing
- Coupled parallel computing

4.1 Distributed Parallel Computing

Under this approach, the task required by a 2GCHAS analysis is distributed over the multiple computer systems on a local area network where 2GCHAS is available. A *client-server* model is used here in the following manner:

A user enters a batch job (typically a multi-case execution scenario) at one host computer (client). The first host would broadcast a service request on the local area network. All hosts on a the local network run a "2GCHAS server process" which listens for requests. Multiple servers may accept a client request based on the current CPU load and other factors.

After receiving responses from each network server, the originating process allocates part of the total batch job up to each server.

The originating process (client) then transmits a copy of the initial input and a subset of the batch job to each server process which has accepted the request. The server

processes then run their part of the job and send the results back to the originating process when it completes.

After receiving the results of one task from a server, the originating client will either schedule another task or close the session.

Advantages:

- Requires minimal coding change to the current 2GCHAS code.
- Makes optimum use of the current hardware resources.

Disadvantages:

- Involves as many data transfers over the net for as there are cases.
- Not suitable for fine grain parallelization.
- Does not provide any improvement for a single case analysis.

4.2 Coupled Parallel Computing

Under this approach, the 2GCHAS analysis code will take advantage of the multiple processors available on a single computer system.

A single job may be executed on multiple processors, with each CPU executing an independent part of the job. For example, multiple element library calls could be performed simultaneously on a multi-processor machine. Communication is through shared memory. The application handles synchronization and scheduling of the independent tasks.

The execution scenario under the Coupled Parallel Computing paradigm will be as follows:

- User enters data in TIP screens (User Input Set or UIS) on a 2GCHAS platform.
- UIS data is saved in the ASCII form of the RDB (Run Data Base).
- The UIS data file is shipped to SGI where the desired analysis is performed, and the analysis results are saved in an ASCII RDB file.
- The ASCII RDB file is transferred back to the originating 2GCHAS platform.
- The data file is restored at the originating platform under the 2GCHAS Executive for the display of analysis results.

Under the Phase I work plan ART recommended ONYX Computer System from SGI as the target machine to demonstrate the feasibility for the application of parallel processing technology in 2GCHAS. The ONYX computer system has 4 high performance processors and provides a rich library of functions for **real-time applications** and **inter-process communication.**

However, 2GCHAS was not available on the SGI computer systems. Consequently, a limited port of 2GCHAS involving the AS, EL, ALIV, LS, and TIP transform units to the SGI environment was required. Essentially, the task was to make the aforementioned units operate in a new RDB environment, which can be ported to another UNIX environment without significant effort. However, port of the input/output portions of 2GCHAS (most of TIP and TOP) was not required, as neither they are in the analysis case loop nor are computationally intensive.

Advantages:

- Potential for performance improvement in a single case analysis.
- Suitable for fine grain parallelization.
- Provides a highly portable 2GCHAS analysis code.
- Involves only one RDB transfers over the net regardless of the number of cases.

Disadvantages:

- Requires substantial coding effort.
- Requires a computer system with at least 2 processors.

The Coupled Parallel Computing approach was pursued during the Phase I effort because it rendered better long term benefits to achieve the overall project goal.

Chapter 5

2GCHAS Analysis Port on SGI

As part of the Phase I effort, a 2GCHAS version 2.3.5 port to the ONYX Computer System from SGI (OS: IRIX 5.2) platform was performed. The following sections describe the tasks performed under this activity.

At the outset, there are 3 programs to perform a desired helicopter analysis, which are listed as follows:

CTRIMW	Performs Trim Analysis
SMANVR	Performs Maneuver Analysis
LSTAB	Performs Linear System Analyses

These programs on SGI are connected to the TIP and TOP portions of the 2GCHAS (running on a SUN e.g.) via the ASCII RDB file. All of the underlying Executive utilities are modified for the desired data access and manipulation. A new RDB approach was developed to remove coupling with the TAE (Transportable Application Executive) software. The following section describes the New RDB approach adopted for the SGI port.

5.1 New RDB Approach

The New RDB approach was developed using the UNIX System V Inter Processing Communication protocols for shared memory allocations. The New RDB approach utilizes the concept of a Mapping table which provides the correlation of RDB data in the shared memory with the Name of the RDB data structure. The structure of the mapping table is given below with some sample examples:

Name	Type	No. of fields /Dimension	Data Type	Size	Offset
MODESC:01	Record	6	xintgr	1	182113
MODESC:02	Record	6	xintgr	1	182114
NSYS	Variable	1	xrealn	1	192119
SYSDOF	Array	1	xchar*44	100	152149
IJ	Array	2	xintgr	100X2	192120

In addition, the various data types, such as integer, real, character and double precision, are kept in separate shared memory segments to avoid data alignment issues while using the XLOCK data access function. The New RDB approach also uses the concept of data volumes for each data type. Each data volume is set to a fixed size (using a parameterized constant), and when more storage is required for that data type, a new volume is created. A master record keeps track of the number of volumes, addresses and shared memory ids for each data type. Even the Mapping table is kept as a separate data volume. A sorted index of the Mapping table entries (RDB Data Names) is kept in a separate volume as well. Mapping table index is sorted (by data names) when a data entry is created or renamed, so there is always a list of sorted indices for RDB data names for each Mapping table volume. As a result, when access to an RDB data item is requested, a binary search for the data name in the sorted Mapping table is performed to obtain the shared memory offset for the data item. This approach resulted in a quick data access.

5.1.1 Advantages of the New RDB Approach

As a result of the aforementioned design features, the New RDB approach presented the following advantages over the current RDB approach:

- The New RDB approach provides **more efficient data access.**
- The New RDB **doesn't require TAE.** TAE itself takes about 16MB of disk storage. TAE is a public domain software distributed by COSMIC.
- The New RDB is **more portable** as it uses the standard UNIX System V protocols for shared memory access. It already works on HP-UX, SUN-Solaris & SGI-IRIX 5.2 platforms.
- The New RDB approach **requires less maintenance** due to drastically smaller number of lines of code:

Partition	Existing Code with Comments	New Code with Comments
EXEC	95021 lines	6010 lines
UTL	37042 lines	11481 lines

- The New RDB approach provides **better implementation on HP** as data items can be directly manipulated in the shared memory without transferring them to the local memory and back to the shared memory. The current HP version of 2GCHAS has severe limitations on the problem size as a data item accessed using the XLOCK function has to be transferred to the local memory from shared memory and vice versa every time the data item is updated.
- The New RDB approach facilitates **direct manipulation of record data structures using the XLOCK function** which was not possible with the existing RDB approach. As a result of this feature, no repackaging of the data in record data structures will be required, and therefore duplication of such data will no longer be necessary. However, this feature has not been exploited in the existing SGI port.
- The New RDB approach presents **no data alignment problem** as various data types are kept in separate shared memory segments.

5.2 Tasks Performed in 2GCHAS Port to SGI

Under the 2GCHAS analysis port to the ONYX computer system, all the procedure and modules at or below the CANALYZ procedure are ported. In addition, some pre- and post-processing modules such as IUISX, CGENARD and APERFC are also ported which are called within the CANALYZ procedure. In particular, the following tasks were performed:

Mapping Table Manager Utilities:

Mapping Table manager utilities are required to maintain RDB data attributes, such as, Data Name, Data Type(integer, real, character etc.), Size, and Shared memory offset. These utilities are used for adding, deleting and accessing an RDB data structure from the Mapping Table.

Replacement for User Language Procedures:

A complete rewrite of the procedures written in the Executive User Language was performed as the Executive User Languate is only supported under the TAE environment which was not ported. In particular the following procedures were replaced by their equivalent FORTRAN/C units.

CMASTER	CANALYZ	CGENARD	CGENSTD	CSETSYS	CSAVSCD
CSAVEFIL	CINCAS	CSETJC	CCASCLR	CLDDATA	

Tools to Compile and Link 2GCHAS Units:

Makefiles were developed to compile and link AS, EL, ALIV, LS, SI and TIP validate & transform units on SGI. In addition, makefiles were developed to prepare libraries of the New Executive and modified Utility functions.

Modification of Executive Data Access and Utility Functions:

All the Executive units are replaced by their equivalent ones to provide the desired RDB data access and manipulations. Currently, 76 2GCHAS Executive Utilities are used in TC4 units, out of which 37 are U*, 21 are X* and 18 are Z* utilities. A detailed list of these function is given below.

1	UABCD	XATDIM	ZALLWS
2	UADDMM	XATFLD	ZATDIM
3	UATBAD	XATSTD	ZATSTD
4	UATBD	XCOPY	ZCOPY
5	UATMPH	XDCLAR	ZDCLAR
6	UCROSD	XDELET	ZRENAM
7	UDDCMP	XEROUT	ZGET
8	UDOTD	XFREWS	ZGETFD
9	UDSLVS	XGET	ZGETSA
10	UDSVDC	XGETFD	ZGETSR
11	UERROR	XGETSA	ZGTWM
12	UEXMPS	XGETSR	ZGTWS
13	UFINDO	XGTWS	ZINIT
14	UINV	XHIER	ZLOCK
15	UINVD	XINIT	ZPUT
16	ULSCLO	XLNGTH	ZPUTSA
17	ULSOPN	XLOCK	ZPUTSR
18	ULSRES	XOPNRO	ZALBWS
19	ULSSAV	XPUTSA	
20	UMOVCD	XRENAM	
21	UMULD	XUNLOK	
22	UMULMM	XWLDCD	
23	UMULMS	XALLWS	
24	UPDATE	XDEFAL	
25	UPROOT	XDELAL	
26	URANK	XPUTFD	
27	URANKD	XPUTSR	
28	USETD	XSTOP	
29	USETI	XWLDCD	
30	USORTN		
31	UTAGJC		
32	UTAGSC		
33	UTRND		
34	UORDRD		
35	UPARSR		
36	USORTD		

5.3 Errors/(Less Desirable Code)Discovered During the Port

The following errors and/or less desirable code was discovered during the 2GCHAS version 2.3.5 port to SGI.

AVWCOF.FOR	EXTERNAL reference to deleted AVWCNR
CUPARD.FOR	Use FORTRAN internal read/write with free format instead of XGT* protocol. Even though the free format internal read/write is said to non-standard, it works on VAX/VMS, HP, SGI, Sun 4.1.2 and Sun Solaris platforms.
CTOPDT.MOD	USCACP is no longer required.
CTRIMW.MOD	Commented UPAUSE
CFCCTL.MOD	Removed GTNCAS and associated logic in the parent unit.
IVSSCO.FOR	UERROR calling sequence in error in DEBUG mode.
IAFOIL.FOR	No initialization of NMACHx NALPHx variables for absent airfoils.
IVALSB.MOD	GUSTOP variable is not initialized.
IVALSM.MOD	IF NCN(i) is zero then *.RN_PS_IDS is not required.
IASCX.FOR	Delete data structures before declaring for repeatability.
LSTAB.MOD	is non-repeatable when CCMG is requested as LCCEA modifies LS_ANALYSIS_OPTION (ARD data) for model type. As a result, number of time steps are set to 1 and number of intervals are set to 0 if number of linearization per period are more than 1.
LRESP.FOR	Removed references to USCxxx
LMSRR.MOD	Removed references to USCxxx
LFR.MOD	Removed references to USCxxx
STATEQ.FOR	Dynamically allocated memory is not released.
SPUDS.FOR	XWLDCD initialization is needed.

Chapter 6

Candidates for Parallelization in 2GCHAS

To maximize performance gain by utilizing the parallel processing technology, the following desirables must be observed in a code:

- *The task to be parallelized must have minimum data dependencies. Any data dependency must be removed for the code to work correctly in parallel.*
- *The time taken in performing the task should be a significant portion of the overall runtime.*
- *The task performed in all paths should be balanced.*
- *The synchronization of the parallel paths should be minimal.*

Based on a careful study of the 2GCHAS System Architecture following the aforementioned guidelines, the following areas were identified as the potential candidates for the application of parallel processing technology in 2GCHAS:

Multicase Analysis:

In order to obtain meaningful analysis outputs, a complex rotorcraft analysis software such as 2GCHAS must be run several times. Currently, 2GCHAS has Multicase analysis capability whereby up to 50 cases of analysis run can be executed from a single program invocation. The user has the freedom to pick and specify the parameters such as aircraft velocity, rotor speed, that may vary from case to case. In a Multicase analysis, typically each case works with a fairly independent data set, and presents an ideal opportunity for parallelization. It is a relatively low risk effort that would provide a high degree of performance improvement.

Linearization:

Linearization analysis, can take advantage of the multiprocessor environment. After Trim or periodic solution analysis, the system equations of motion are in second order periodic coefficient form. To perform a linear system analysis, linearization of the second order periodic system is typically performed. The linearization analysis of a second order periodic system requires linearizing matrices and force vectors of the periodic system by perturbing each state at each azimuthal location (see 6.4 for details) and computing partial derivatives of the force vector. Each state perturbation contributes to a separate column to the linearized matrices. Therefore, there is no data dependency in computing partial derivatives of the force vector with respect to one state with computing partial derivatives of the force vector with respect to another state. Similarly, there is no data dependency in computing linearized matrices at one azimuth, with computing linearized matrices at another azimuth. Therefore, parallelization in the linearization analysis can be achieved both at the state perturbation level as well as at the azimuthal level.

Assembly of Elements:

In performing either a trim, maneuver or linearization in stability analysis, assembly of equations of motions is required. The matrices of all the elements of the model are calculated and then they are assembled in system matrices. The assembly process takes a significant amount of CPU time as it is nested inside the trim convergence, periodicity convergence and Newton-Raphson iteration loops. (See Figure 6.1, 6.2 and 6.3 for details). Any improvement in the performance of the assembly process would result in significant improvement in Trim, Maneuver and Stability analysis. Since each element matrix computation works only on the given element data, there is no data dependency among the elements. Therefore, the assembly process is quite amenable to parallelization.

Airloads Computations:

The airload and induced velocity computations are performed on each aerodynamic computation point (ACP) and are passed back to the corresponding finite element. On a given iteration, the airloads and induced velocities at each ACP are independent and can therefore be computed in parallel.

Vortex Wake Computations:

Vortex wake model in rotorcraft analysis is a computationally intensive process. The reason is that for computing the induced velocity at one particular collocation point in space, the Biot-Savart Law computations has to be performed for all the wake elements. At each time step in the prescribed vortex wake model, this process is repeated for all the aerodynamic control points (ACPs) on the rotor blade to calculate the rotor inflow. For example, assuming a 4-bladed rotor with 10 ACPs in each blade, if 4 turns of wake elements are considered where each turn contains 24 elements, the number of the Biot-Savart computation set for the 3-dimensional rotor inflow is $3 \times 4 \times 10 \times 4 \times 24 = 11,520$.

If the free vortex wake is chosen, the process is even more expensive since the induced velocity is needed for every wake elements also. This means that the additional number of the Biot-Savart computation set for free wake is proportional to the square of number of wake elements. For the same rotor, the additional computation in free wake is $3 \times 4 \times 24 \times 4 \times 24 = 27,648$.

In total, for the above mentioned example, it would require 39,168 computation sets at each time step. To reduce the computational effect, vortex wake model usually assumes the wake geometry is frozen and only the vortex strength is changed during one rotor revolution. The free wake geometry and a geometry influence coefficient matrix involving the CPU intensive Biot-Savart integration are computed only at each beginning of rotor revolution. This assumption makes the scheme well structured.

In the vortex wake computations, the outermost loop is on the number of azimuth steps in one revolution. Inside this loop, the process is the same and there is no data dependency. Clearly, this is the ideal place for the application of parallel processing in the vortex wake computations. The number of azimuth steps in one revolution is typically a round number, such as, 24, 36, 48, ... etc. Therefore, processors can be loaded evenly for the vortex wake computation at the azimuth level. Since it is the outermost loop the vortex wake computations, the performance gain with the parallel processing should be significant.

Equation of Motion Solution:

In performing either a trim, maneuver or linearization in stability analysis, solution of equations of motions is required. The solution process takes a significant amount of CPU time as it is nested inside the trim convergence, periodicity convergence and Newton-Raphson iteration loops. (See Figure 6.1, and 6.2 for details). Any improvement in the performance of the solution process would result in significant improvement in Trim, Maneuver and Stability analysis.

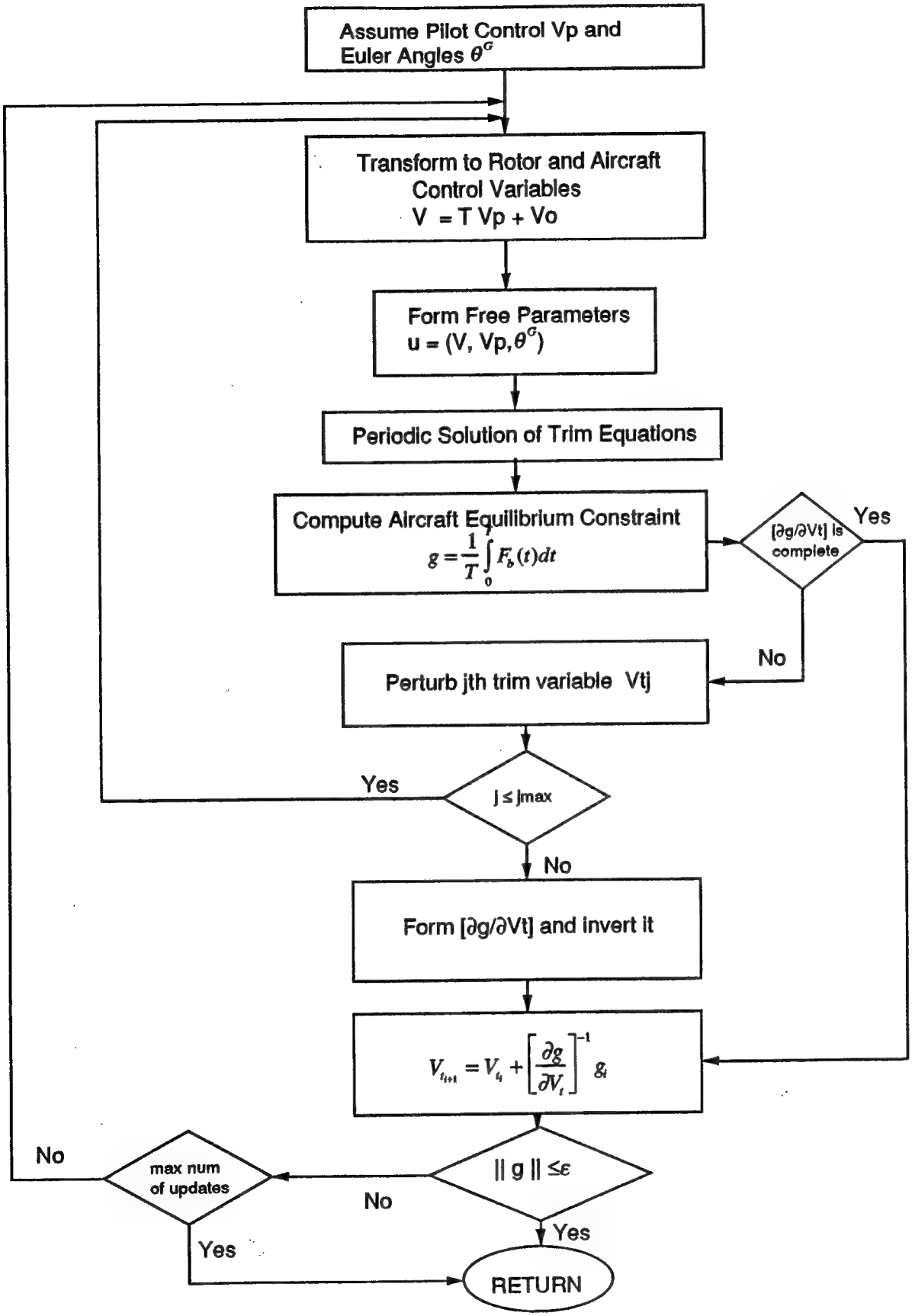


Figure 6.1: Flow Chart for Trim Analysis

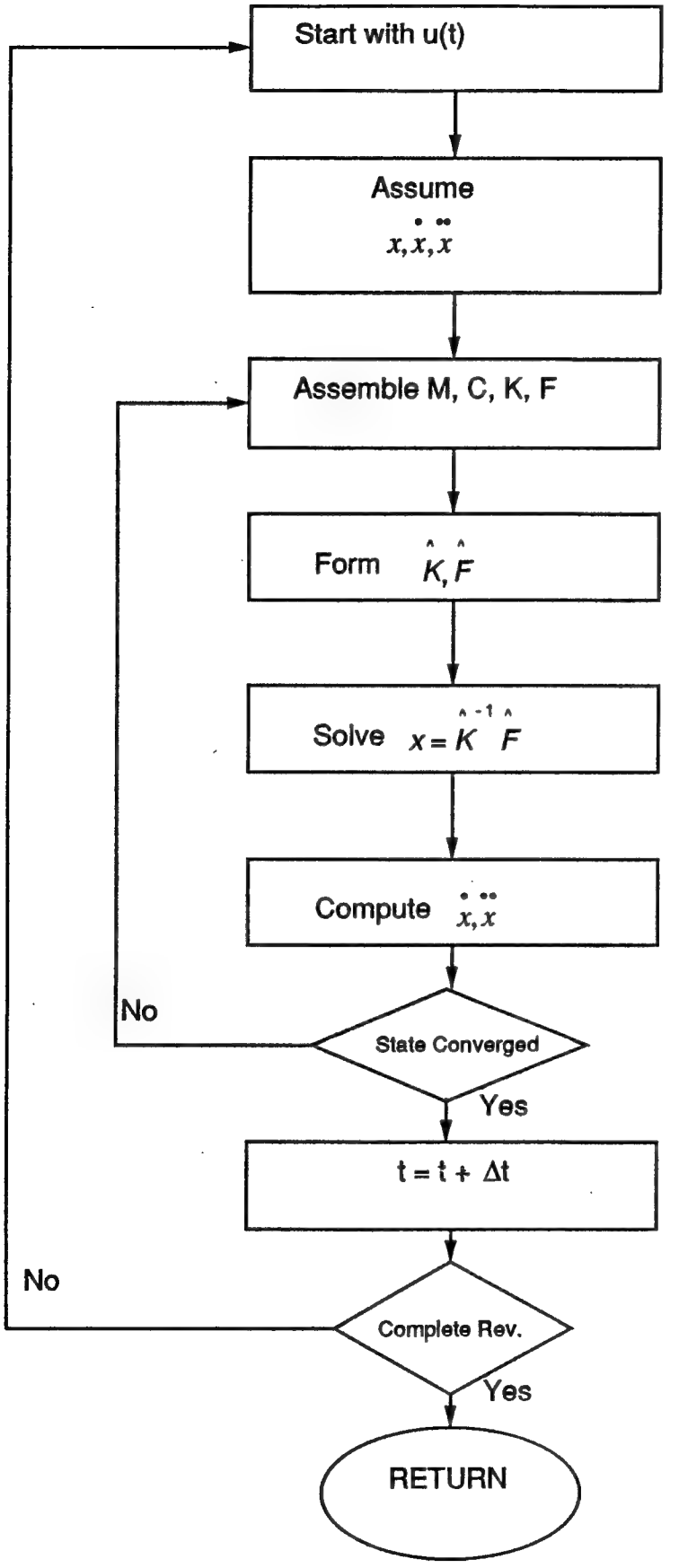


Figure 6.2: Flow Chart for Periodic Solution

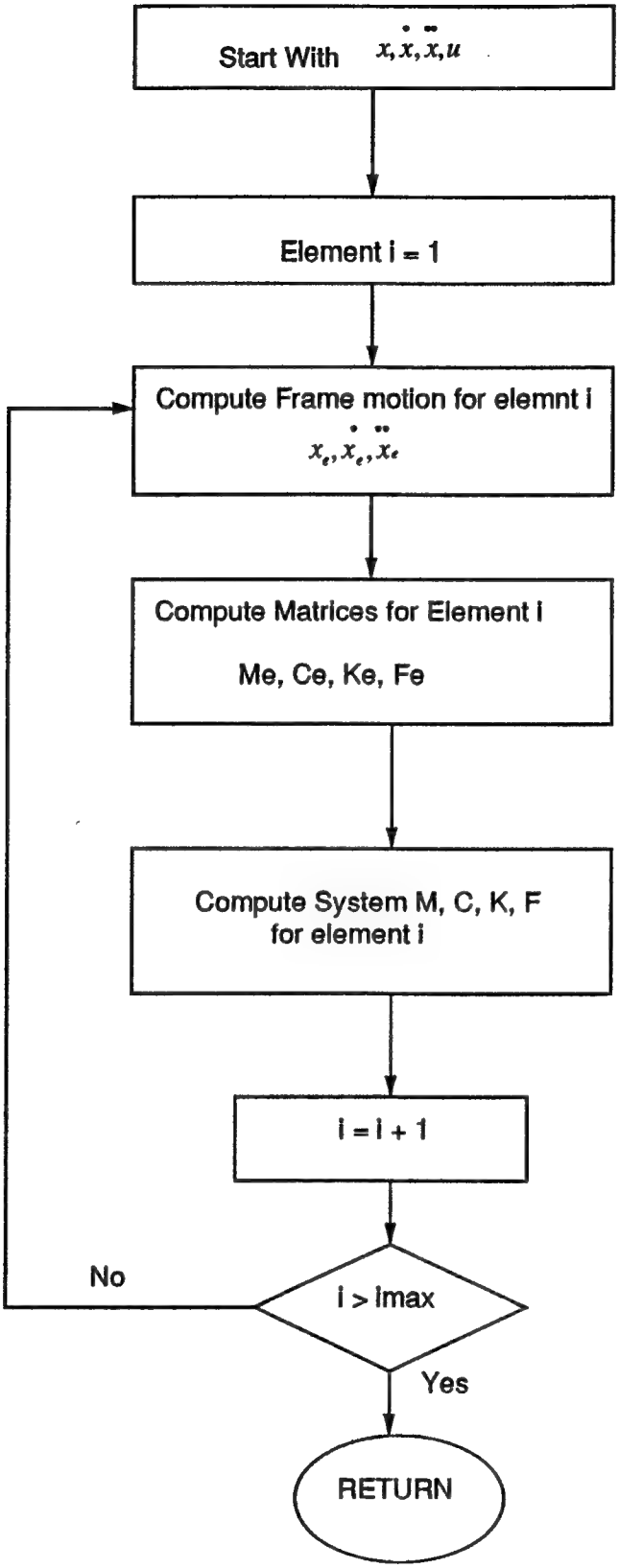


Figure 6.3: Flow Chart for Assembly of Equations of Motion

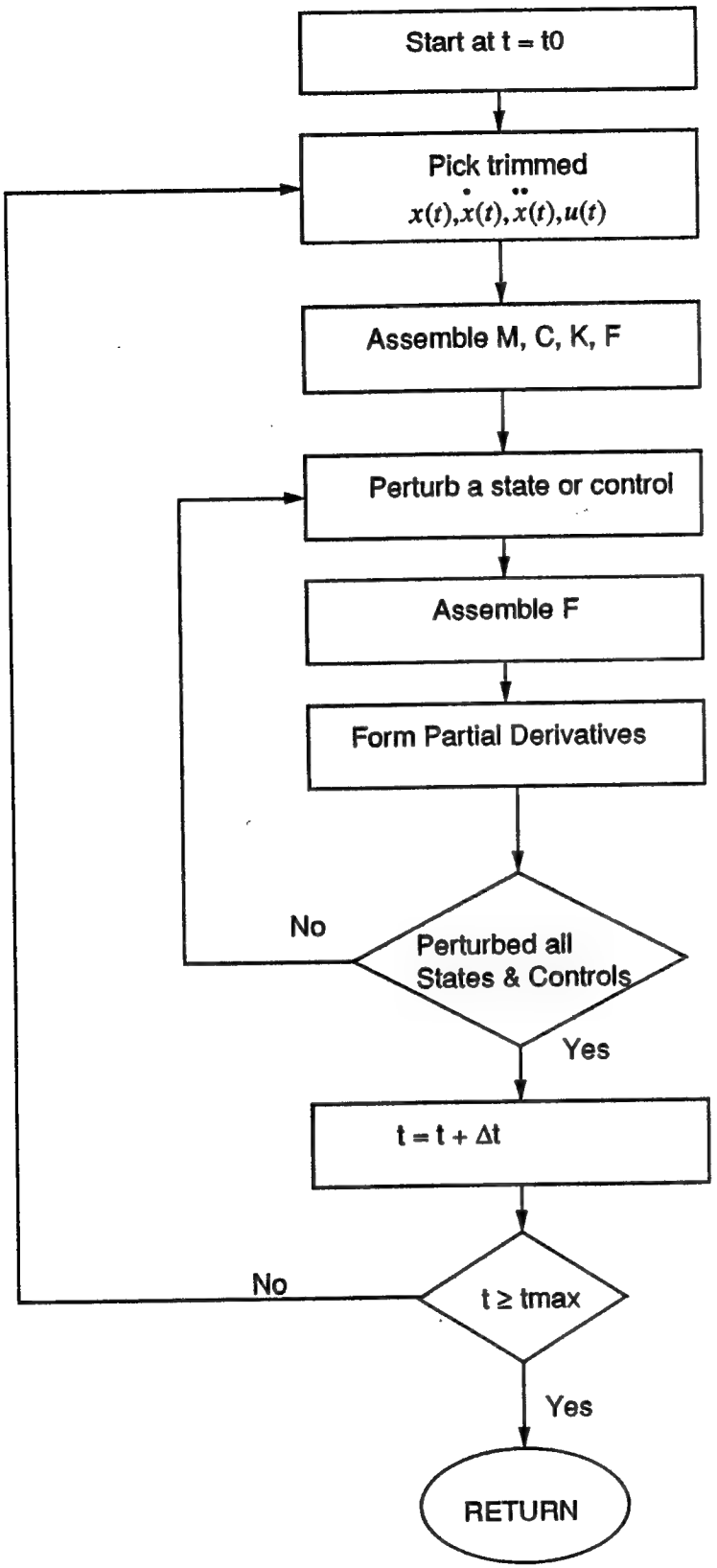


Figure 6.4: Flow Chart for Linearization Analysis

Chapter 7

Parallelization in 2GCHAS

7.1 Multicase Analysis Parallelization

Parallelization in 2GCHAS was first attempted at the multicase analysis level as it provided the most attractive opportunity to utilize the parallel processing technology. The multicase analysis loop is the outermost loop 2GCHAS which is performed under the top level unit CMASTER. The multicase eigenanalysis of the Bearingless Main Rotor (BMR) of the Comanche Helicopter 1/6th scale model, was used as the sample problem to test the multicase analysis parallelization. The following section provides a brief description of the tasks performed to achieve the parallelization at the multicase analysis level:

7.1.1 RDB Copy Function

A separate copy of the RDB is required for each thread in the multicase analysis because the analysis software reads and writes various data structures in the RDB. As a result, function **xcprdb** was written to make a copy of the RDB for a given thread. This function, allocates the desired shared memory segments for the new RDB and also copies the data from the source RDB.

7.1.2 Parallelization Functions

The parallelization paradigm used here requires that the master process spawns as many slave processes as required and waits until all the slave processes are finished. To achieve, this task following functions were written using the System V inter-process communication protocols.

xgtnp: This function returns the number of processor to be used (or number of slave threads to be generated). The function gets the value from the environment variable MP_SET_NUMTHREADS if it is defined, otherwise it assumes 1 processor and returns 1.

spinit: This function sets and holds signal SIGCLD, which is used to tell if a slave process is finished. In addition, it attaches signal SIGCLD with the **handler** function.

submit: This function creates a slave process using the **execl** System V protocol. It also submits the required command in the slave process, and increments the number of slave processes count.

handler: This function is triggered when the SIGCLD signal is received. It uses the **wait** protocol to receive the slave process termination status. This function also reduces the number of slave processes remaining, and resets signal SIGCLD with the **handler** function attached to it.

npblock: This function waits until all slave processes are completed.

7.1.3 Modification to Existing 2GCHAS code

The modification to the existing 2GCHAS code was confined to the following top level routines:

1. Main program CMASTER.F is modified. Figure 7.1 outlines the process description of this subroutine.

2. A new subroutine ANALYZ.F was created which performs its share of analysis cases. It gets the processor number and the total number of available processors as input. Figure 7.2 shows the process description of this subroutine. A corresponding **canlyz.c** was created to obtain the required arguments from the command line.

3. A new subroutine MPSAVE.F was created which saves the SC data after the analysis is completed. A corresponding **csavsc.c** unit was created to obtain the required arguments from the command line.

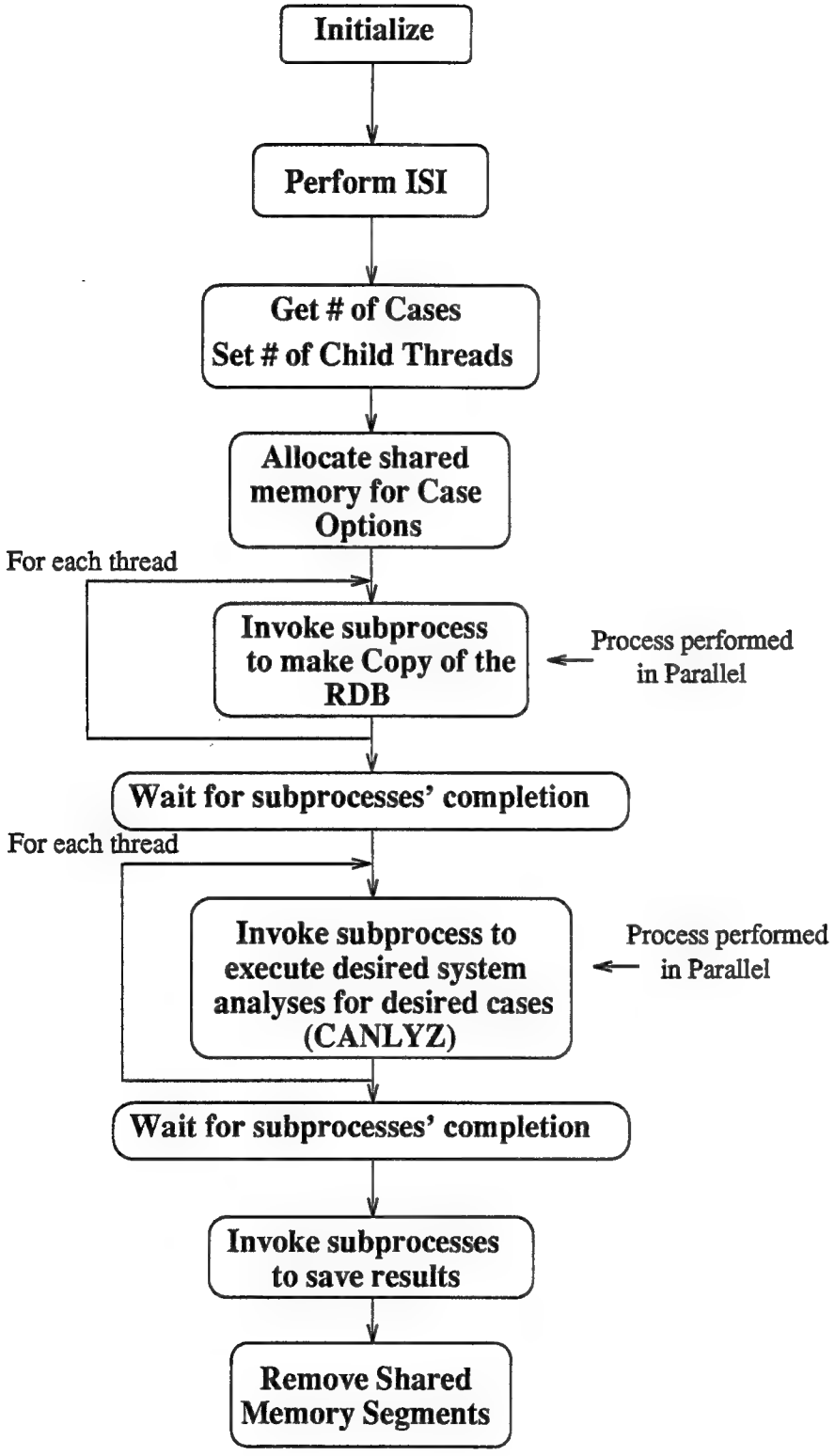


Figure 7.1: Overall Process Control in 2GCHAS (Parallel Version)

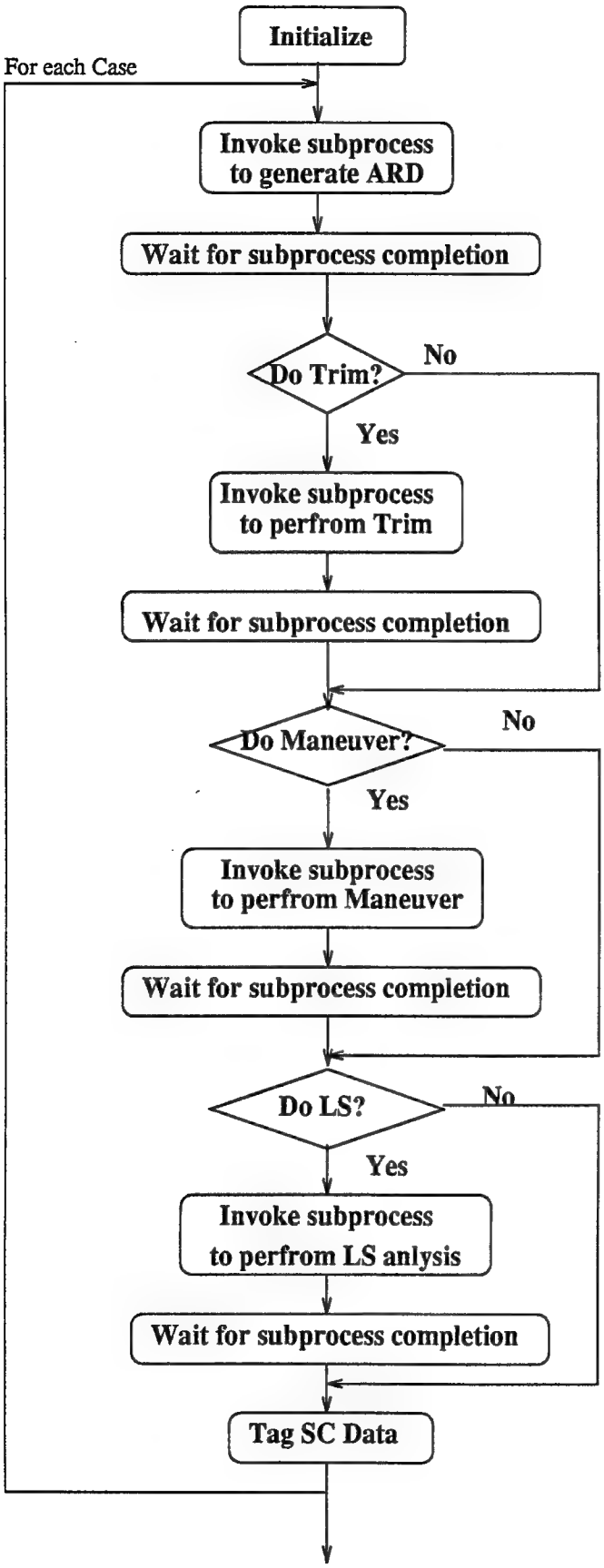


Figure 7.2: Flow Chart for System Analyses in 2GCHAS (CANLYZ)

7.2 Parallelization in Single Case Analysis

To choose an appropriate candidate for parallelization in a single case analysis, a timing study was conducted to analyze the most time consuming areas in 2GCHAS. Appendix A shows the results of the timing study. Of the three problems studied, the eigenanalysis of BMR rotor was selected as the sample problem to take advantage of the parallel processing technology. However, the problem demonstrated that the majority of the overall runtime was consumed by Eigenanalysis (52%) and Linearization (31%). Therefore, optimization of the eigenanalysis algorithm became the next order of business. The following efforts were made to improve the eigenanalysis solution algorithm.

7.2.1 Eigenanalysis Algorithm Improvements

To reduce the time taken in performing a eigensolution, the following eigenanalysis solution methods were attempted:

Arnoldi Eigensolver

This method takes advantage of the fact that only a subspace of the overall system is solved to obtained the desired subset of the eigenvalues. A public domain software, developed at the Rice University in Texas, was used to verify the validity of the method. The results obtained for a few eigenvalues gave the highest eigenvalue modes instead of the lowest eigenvalue modes typically desired. As a result, the Shift-Inverse eigenanalysis problem was attempted. However, the results obtained from the Shift-Inverse eigenanalysis were incorrect due to ill-conditioning of the resulting matrices. There was also an issue of convergence with this approach, as it is an iterative method.

LAPACK Eigensolver

LAPACK is a transportable library of Fortran 77 subroutines for solving the most common problems in numerical linear algebra: systems of linear equations, linear least squares problems, eigenvalue problems, and singular value problems. It has been designed to be efficient on a wide range of modern high-performance computers. LAPACK is intended to be the successor to LINPACK and EISPACK.

DGEEVX eigensolution routine was used from the LAPACK software. This routine computes the eigenvalues and left and right eigenvectors of a general matrix, with preliminary balancing of the matrix, and computes reciprocal condition numbers for the eigenvalues and right eigenvectors. This eigensolution method was used to solve various 2GCHAS eigenanalysis problems.

The results obtained using this software matched very well with the existing 2GCHAS results. On an average this method is about **4 times faster** than the PVA (Principal Vector Algorithm) used currently in 2GCHAS. Consequently, this method was

implemented in the SGI version of 2GCHAS.

At this point, the linearization analysis became the most time consuming portion in the BMR eigenanalysis problem. Therefore, parallelization of the linearization analysis was attempted next. The following section describes the efforts taken to parallelize the linearization analysis at the azimuth level.

7.3 Parallelization of Linearization Analysis at Azimuth Level

For a second order periodic system, the linearization analysis involves computing the linear matrices at several azimuths. Figure 6.4 shows the linearization analysis flow chart. The linearization analysis is specially a very time consuming process for a large periodic system which involves significant hub motion as it requires obtaining the linear matrices at several azimuths. The following efforts were made to parallelize the linearization analysis at the azimuth level.

1. Subroutine LSTAB was modified. Figure 7.3 outlines the process description of this subroutine. This routine create RDB copies for each thread, and invokes LSMCK routine (as slave process) to compute linear matrices. The parallelization routines developed duing the multicase analysis parallelization were used here as well to perform the aforementioned task. Once all the linear matrices are computed for all azimuths, it loops over all the RDBs and accumulates the linear matrices. It finally divides all the linear matrices by the number of azimuths to obtain their average value used for further Linear System analysis.

2. A new subroutine LSMCK.F was created which performs its share of linear matrices generation. It gets the processor number and the total number of available processors as input. Figure 7.4 shows the process description of this subroutine. A corresponding **clsmck.c** unit was created to obtain the required arguments from the command line.

3. Routine LGCCM.F was modified such that it only performs the linear matrices addition. Earlier this routine was also responsible for creating the linear matrices at the first azimuth, and for computing the average at the final azimuth.

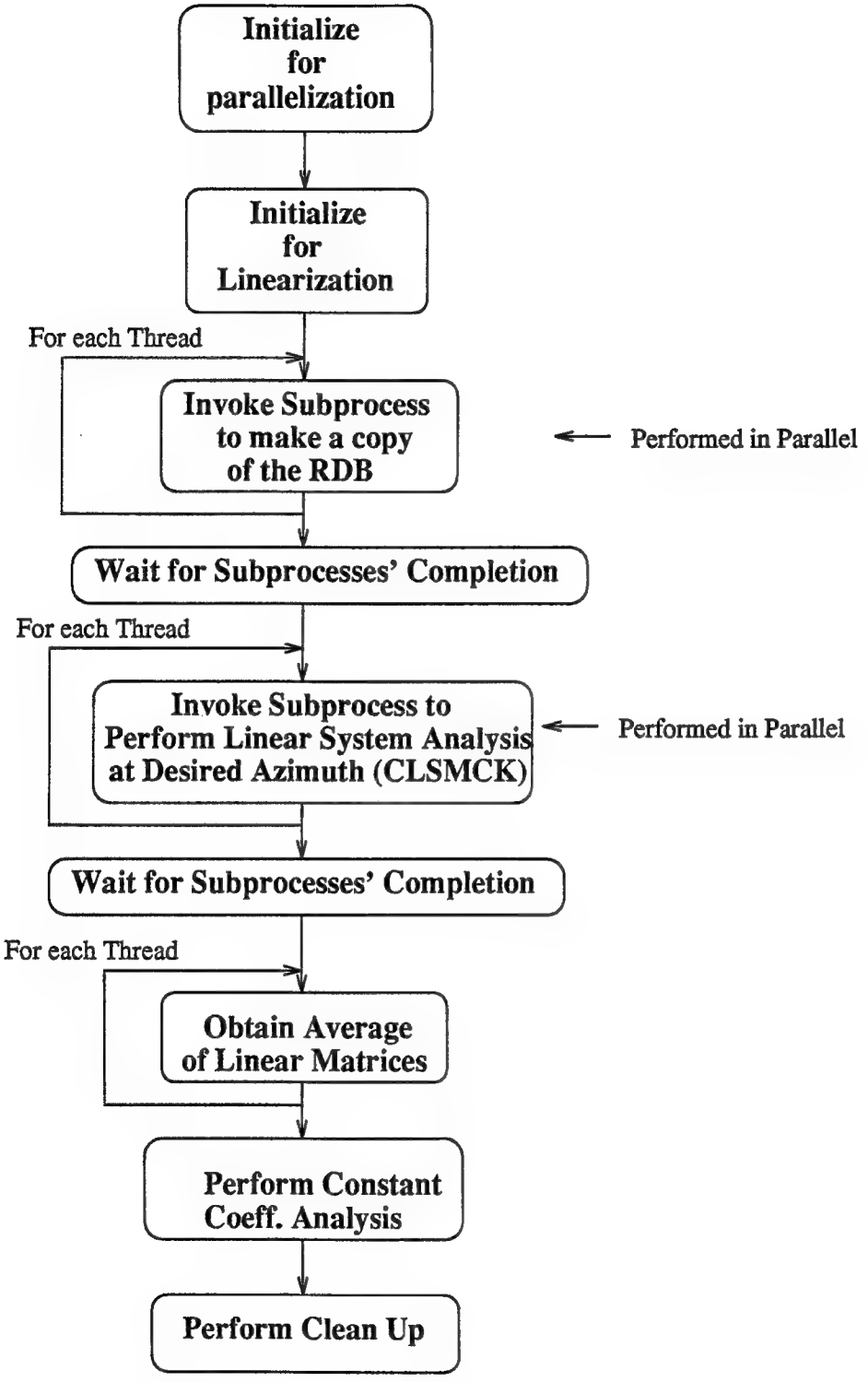


Figure 7.3: Process Control for Linear System Analysis (LSTAB Parallel version)

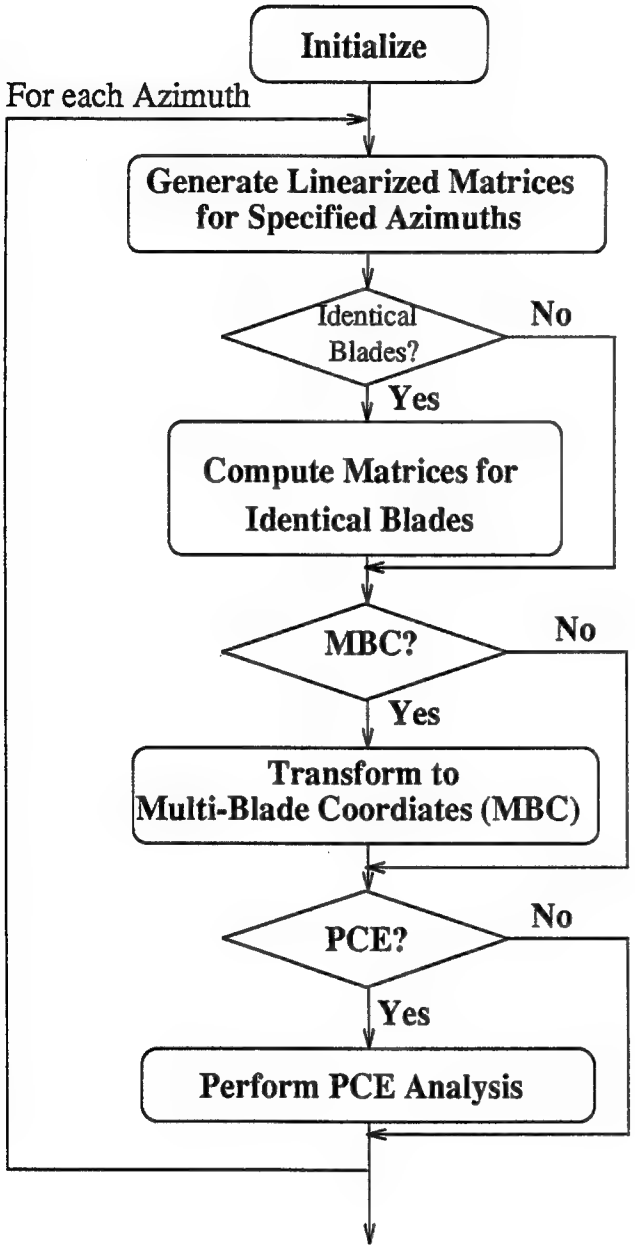


Figure 7.4: Flow Chart for Linear System Analysis at Azimuth Level(LSMCK)

Chapter 8

Results

8.1 Results from the 2GCHAS Port to SGI

After the 2GCHAS port to the SGI platform, several 2GCHAS system test problems were run to test the various analysis functionality in 2GCHAS. Table 1 gives a comparison of the 2GCHAS performance Sun-Sparc2 platform and the ported 2GCHAS performance on SGI-ONYX, along with the functionality tested. The 2GCHAS version on SGI was also single threaded.

The SGI machine CPU was found to be about 3 times faster than SUN's Sparc-10 CPU. The additional improvement is primarily due to the **New RDB approach** and code optimization.

In addition, several other 2GCHAS test problems were run to validate the port to SGI platform. The overall time reduction factor was found to be between 5 to 8 for single case analysis problems and between 7 to 10 for the multicase analysis problems. The additional gain in multicase analysis was attributed to less overhead involved in spawning a sub-process. Currently, the proc invocation in the existing 2GCHAS version is fairly time consuming, and becomes significant in a multicase analysis.

Table 1: Study of 2GCHAS Performance on SGI vs. SUN

Problem	SS10 sec.	ONYX sec.	Factor $\frac{ONYX}{SS10}$ CPU	Gain from Port	Total Gain	Tested Functionality
p_04	144	28	3	1.71	5.34	Periodic Solution
p_07	1500	216	3	2.31	6.94	Vortex Wake
p_17	3660	475	3	2.56	7.7	Trim+Lin.Trans.Resp.
pn5_s1b	small	small	3	–	–	Maneuver
pn1_s2	small	small	3	–	–	Eigenanalysis

Table 2: Timing Study for BMR Eigenanalysis with Multicase Analysis in Parallel

Sun SS10 Sec.	Factor $\frac{ONYX}{SS10}$ CPU	Expected Runtime on SGI ONYX Sec.	Actual Runtime w/ port to SGI Sec.	Gain from port + code opt.	with new eig. sol. Sec.	Gain From new eig. sol.	with 3 CPU Sec.	Gain from 3 CPU	Total Gain Over SS10
BMR Eigenanalysis (single case)									
4560	3	1520	561	2.7	417	1.35	–	–	10.93
BMR Eigenanalysis (12 cases)									
55162	3	18387	5125	3.58	3810	1.35	1737	2.2	31.75
P_13 Linearization + MBC Transformation + eigenanalysis 12 cases									
2206	3	735	244	3.01	206	1.18	87	2.37	25.35

8.2 Results of Parallelization in the Multicase Analysis

After the parallelization was achieved at the multicase analysis level, several existing 2GCHAS system tests involving multicase were run to validate the parallel version of 2GCHAS. This version of 2GCHAS also took advantage of the code optimization compiler option on SGI and the new Eigensolver from the LAPACK software package. The results obtained were identical to the single processor version, which was earlier validated.

In addition, a timing study was performed to measure the gains from various sources of performance improvement. Structural dynamic (eigenanalysis in vacuo) of a coupled rotor-fuselage system with multi-blade coordinates (P_13 12 cases), and the eigenanalysis of the Bearingless Main Rotor (BMR) of the Comanche Helicopter 1/6th scale model (12 cases), were chosen as the sample problems to measure the performance gain due to parallelization in the multicase analysis.

The timing study findings are given in table 2.

The additional gain in the single processor/single case version comes from the faster eigensolution method. The multicase version shows even a greater performance gain due to less overheads in sub-process invocation. The 3 CPU version shows further improvement but doesn't show a linear gain in performance due to the following reasons:

First of all, for optimum performance on a N processor machine, the maximum number of slave threads to be used must be N-1, as the system always needs at least 1 processor to perform operating system related tasks. In addition, the master thread also needs some resources to perform the slave processes synchronization task. The net gain with the additional CPUs is also not linear, which was confirmed by a study on a small CPU intensive program. The performance gain in this example with 3

Table 3: Timing Study for P_45a problem with Linearization Analysis in Parallel

Sun SS10 Sec.	Factor $\frac{ONYX}{SS10}$ CPU	Expected Runtime on SGI ONYX Sec.	Actual Runtime w/ port to SGI Sec.	Gain from port + code opt.	with new eig. sol. Sec.	Gain From new eig. sol.	with 3 CPU Sec.	Gain from 3 CPU	Total Gain Over SS10
7769	3	2590	1454	1.78	1406	1.03	702	2	11.06
Linearization only									
6560	3	2187	1188	1.84	1188	1.0	469	2.53	14

processor was about 2.6. The performance gain is further reduced if any of the processes gets involve in page faulting or memory swapping, or perform I/O operation as there is only one disk controller.

In case of the BMR eigenanalysis the size of the RDB for single thread is about 45MB. Since there are 3 threads, and there is only 96MB of memory available on the computer system, it spends time in doing page faults or memory swapping. As a result, the overall performance gain is reduced.

In case of the second example, out of 87 seconds, 10 seconds were taken by the RDB copy operation. Consequently, only 77 seconds were actually spent in doing the computations. Therefore the true gain due to parallelization is 2.67, which is the maximum gain observed with 3 processors.

8.3 Results of Parallelization in the Single Case Analysis

Linearization analysis was parallelized at the azimuth level to reduce the runtime in a single case analysis. 2GCHAS test problem, P_45a was modified to obtain a periodic solution at 160 knots before performing linearization at every azimuth. In addition, the number of azimuths per period were increased to 72 from 24. The results obtained were identical to the single processor version, which was earlier validated.

In addition, a timing study was performed to measure the overall performance gain from the parallelization in the linearization analysis at the azimuth level.

The timing study findings are given in Table 3.

Chapter 9

Conclusion - Technical Feasibility and Future Work

9.1 Conclusions

The goal of the Phase I effort was to define the specification for the parallelization opportunities in 2GCHAS, and apply the parallel processing techniques on a sample analysis to demonstrate a proof of concept. The tasks performed during the Phase I Base effort went beyond meeting the Phase I objectives and included the following:

- Port of the most recent version of 2GCHAS Analysis on the Silicon Graphics Computer System.
- Parallelize multicase analysis in 2GCHAS.
- Choose an appropriate analysis component to demonstrate application of parallel processing technology in a single case analysis.
- Parallelize Linearization Analysis and thereby reduce the overall runtime of a single case Linear System analysis.
- Identify candidates for future parallelization in 2GCHAS.

In addition, efforts had been made to improve the eigenanalysis solution method.

The Phase I effort not only demonstrated the technical feasibility of the parallel processing techniques application to a complex helicopter analysis software, but also rendered a port of a significant portion of 2GCHAS Software which will provide a platform to apply the parallel processing techniques to the other candidate areas in the 2GCHAS Software during the Phase II effort.

9.2 Phase II Justification

The results obtained from the Phase I effort demonstrated over an order of magnitude improvement in the overall performance for the sample problems due to the following reasons:

- **Parallelization**
- **Better and more portable RDB implementation**
- **Less overhead in a process invocation**
- **Improved eigenanalysis algorithm**
- **Port to an SGI platform.**

The Phase I results indicate that there is a tremendous potential in improving the overall performance of 2GCHAS by the application of Parallel Processing Technology, New RDB Approach, better solution techniques and a port to the SGI platform. The resulting performance gain will substantially benefit the 2GCHAS user community, as the problems which earlier took hours to run, would then be run in a matter of a few minutes. The reduced turn-around time will significantly improve the effectiveness of 2GCHAS, and will be sincerely appreciated by the helicopter industry.

Computationally intensive applications are often tailered to take advantage of a specific computer architecture to expedite the overall runtime. For example, Massively Parallel computer systems using Single Instruction Multiple Data (SIMD) architecture have been utilized to improve the performance of the Computational Fluid Dynamics(CFD), Acoustics and Electro-Magnetic applications. These SIMD computer systems perform the same instruction on various processors with different data sets. While this feature is very appropriate for certain applications, it also limits the type of applications that can be efficiently run on such a computer system. The Distributed Parallel computing paradigm presented here can be expanded to run different portions of an application over a network on different computer systems with the computer architecture suited for the respective portion of the application.

Under a Phase II SBIR, ART is integrating Computational Structural Dynamics (CSD) and the Computational Fluid Dynamics (CFD) codes. The integration task addresses the analytical coupling of the CFD and CSD codes, and will require porting of the CFD software on a conventional computer system. However, the performance issues of the resulting analysis code will still need to be addressed as the CFD codes are far more computationally intensive than the CSD codes. Therefore, a suitable parallelization technique will be required to lower the overall runtime in a CSD plus CFD analysis. CFD codes are typically available on a massively parallel, single instruction multiple-data (SIMD) computer system, while the CSD applications are

not suited for such a computer architecture. During the Phase I effort, Distributed Parallel computing approach was presented which can be applied to perform CFD code on a Massively Parallel, SIMD computer system while performing CSD code on a conventional workstation. Fortunately, the data communication over the network will not be an issue as the interface data between CFD and CSD application is relatively small.

Computational intensity in a CFD analysis comes from the facts that in a typical model a large number of the aero grid points, which is in the order of 1,000,000, are required, and a solution to a system of partial differential equations is obtained iteratively on each grid point. In a typical rotorcraft model, such a model grid has to be analyzed for each blade independently. However, due to the overwhelming computational and memory requirements only a single blade is typically analyzed. Distributed parallel computing can alleviate the computational and memory limitations presented in such an application, and therefore make it possible to perform CFD analysis of a full rotor system where each blade is analyzed in parallel on a separate computer system.

The overall Phase II effort, will therefore, not only improve performance of the CSD application on a specific computer system, but will provide an unprecedented environment via Distributed Parallel computing to couple related applications such as CFD or Acoustics analysis to the CSD application to render optimal performance.

9.3 Future Work - Phase II

9.3.1 Coupled Parallel Computing

Using the approach developed during the Phase I effort, the primary objectives for the Phase II will be detailed design, implementation and testing tasks in the following areas:

1. Complete 2GCHAS Port to the SGI Platform. This port will also include a port of the TIP Kernel for data acquisition screens, dynamic tutor for output selection, and Output utilities of TOP. The resulting 2GCHAS port will be stand-alone which will no longer require TAE.

2. Improve 2GCHAS code by utilizing the New RDB features. Currently, a significant amount of data repackaging is performed to transfer data from the record data structures to the data structures that can be directly manipulated by using the XLOCK function. This process not only takes time but also consumes shared memory resources in keeping another copy of the ARD data. This step will no longer be required in the New RDB approach, as record data structures can be directly manipulated using the XLOCK function.

3. Improve 2GCHAS solution techniques by utilizing better, faster and parallel solution algorithms.

4. Apply parallel processing paradigm to other candidate areas in the 2GCHAS Software which were identified during the Phase I effort.

5. Develop techniques to maximize performance gain for a given problem. Currently, the performance improvement due to parallelization in a single case analysis was confined to only a class of problems. However, when other areas in 2GCHAS will be parallelized, techniques will be required to apply parallelization at a level, which will result in maximum performance gain for a given problem.

6. Validate 2GCHAS on the SGI Platform using the existing 2GCHAS system tests.

7. Perform a timing study to document the gain in the overall performance of 2GCHAS.

8. Prepare addendum to 2GCHAS User Manual, Application Manual, and Programmer Manual for the Parallel version of 2GCHAS.

9.3.2 Distributed Parallel Computing

In addition to the Coupled Parallel Computing Tasks, the Distributed Parallel Computing approach will be applied in Phase II to make the most use of the hardware resources that are available over the network. This approach will be particularly useful in integrating the Computational Fluid Dynamics (CFD) codes with Computational Structural Dynamics (CSD) codes. ART is currently working on a Phase II SBIR topic to develop an Advance Computational Fluid Dynamics Methods for Elastic Helicopter Blades. However, since the CFD codes are far more computationally intensive than the CSD codes, Distributed Parallel Computing will be imminently required to obtain a reasonable turn-around time on a CSD plus CFD analysis.

The main objective here will be to develop an environment that can provide flexibility to run different pieces of a software on a the computer system that is most suited for the portion of the application with minimal existing code modification. The detailed task are as follows:

1. Developing a scheduling negotiation network protocol. This will be based on TCP/IP, UDP/IP, Sun RPC/XDR, or the ISO/OSI protocol suite, depending on which protocol is determined to be the most suitable.

   This protocol will include facilities for server allocation and load balancing as well as basic job scheduling.

2. Developing a transport protocol for interface data elements. This may be as simple as transporting the ASCII data encoded over a bytestream protocol.

3. Developing a process executive to control execution of the batch jobs independently of the 2GCHAS executive. This will be done to provide process execution on other operating systems, and to facilitate augmenting CSD codes such as 2GCHAS with specialized applications that may run on custom computer architectures.

Appendix A

Results of the Timing Study on Typical Helicopter Analysis

The timing study was performed using the 2GCHAS version 2.3.5 on the Onyx Computer System using OS IRIS 5.2.

A.1 Trim analysis of the Uh60 model with Free-wake

Number of Dof = 212
Total time = 1680 sec

Analysis	Time (sec)	Remark
Trim	804	with Uniform Inflow
Trim	434	with Prescribed Wake
Trim	402	with Free Wake

A.1.1 Uniform Inflow Computation Breakdown

Analysis	Time (sec)	Remark
Assembly	1.022	First time step/first iter.
Solution	.507	First time step / first iter.
Assembly	.491	Subsequent time step/first iter.
Solution	.507	Subsequent time step/first iter.
Assembly	.334	Subsequent time step/subsequent iter.
Solution	.507	Subsequent time step/subsequent iter.

A.1.2 Prescribed Wake Computation Breakdown

Analysis	Time (sec)	Remark
Assembly	8.912	First time step/first iter.
Solution	.512	First time step / first iter.
Assembly	.637	Subsequent time step/first iter.
Solution	.512	Subsequent time step/first iter.
Assembly	.478	Subsequent time step/subsequent iter.
Solution	.512	Subsequent time step/subsequent iter.

A.1.3 Free Wake Computation Breakdown

Analysis	Time (sec)	Remark
Assembly	59.534	First time step/first iter.
Solution	.508	First time step / first iter.
Assembly	.623	Subsequent time step/first iter.
Solution	.508	Subsequent time step/first iter.
Assembly	.468	Subsequent time step/subsequent iter.
Solution	.508	Subsequent time step/subsequent iter.

A.2 Eigenanalysis of a BMR Rotor-Fuselage Configuration

Number of Dof = 236
Total time = 561 sec

Analysis	Time (sec)
Static Eqbm.	49.881
Linearization	176.154
Eigenanalysis	293.103

A.3 Nonlinear Transient Response of a BMR Rotor-Fuselage Configuration

Number of Time steps = 500
Number of Dof = 236
Total time = 1680 sec (with uniform inflow, and sparse matrix solution)

Analysis	Time (sec)	Remark
Assembly	1.022	First time step/first iter.
Solution	.235	First time step / first iter.
Assembly	.815	Subsequent time step/first iter.
Solution	.215	Subsequent time step/first iter.
Assembly	.515	Subsequent time step/subsequent iter.
Solution	.215	Subsequent time step/subsequent iter.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	9 December 1994	Final Report 7/14/94 to 12/9/94

4. TITLE AND SUBTITLE
Application of Parallel Processing Technology in Complex Helicopter Analysis

5. FUNDING NUMBERS
NAS2-14084

6. AUTHOR(S)
Dharmendra Kumar - Principal Investigator
Hossein Saberi - Project Manager

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Advanced Rotorcraft Technology, Inc.
1685 Plymouth Street, Suite 250
Mountain View, CA 94043

8. PERFORMING ORGANIZATION REPORT NUMBER
FR7060-001

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NASA Ames Research Center
Attn: Dr. Michael Rutkowski, M/S 215-1
Moffett Field, CA 94035-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NAS2-14084

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unlimited/unrestricted

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Complex Helicopter Analysis codes, such as Second Generation Comprehensive Helicopter Analysis System (2GCHAS), have been developed to provide a common framework for interdisciplinary design and analysis of rotorcraft systems. However, a second generation comprehensive analysis code's effectiveness is limited by the slow processing speed due to the computational intensity of the software. The goal of Phase I effort is to define the specification for the parallelization opportunities in 2GCHAS, and apply the parallel processing techniques on a sample analysis to demonstrate a proof of concept. In particular, Phase I tasks included: 1) Port the latest version of 2GCHAS to the Silicon Graphics' ONYX Computer System. 2) Parallelize multicase analysis in 2GCHAS. 3) Choose an appropriate analysis component for parallelization. 4) Parallelize selected analysis component. 5) Identify other candidates for parallelization. Phase I results showed over an order of magnitude reduction in the overall runtime for the sample problems due to parallelization, better Run Data Base implementation, improved analysis algorithm and faster machine speed. Moreover, Distributed parallel computing approach, presented during the Phase I, will provide an environment to couple other related applications such as Computational Fluid Dynamics with the Computational Structural Dynamics codes such as 2GCHAS to obtain optimal performance.

14. SUBJECT TERMS
Parallel Processing, Complex Helicopter Analysis, Computational Structural Dynamics, Computational Fluid Dynamics, Coupled Parallel Computing, Distributed Parallel Computing

15. NUMBER OF PAGES
46

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIED	UNLIMITED